2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2025)
October 19-25, 2025. Hangzhou, China

2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3315-4393-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/IROS60139.2025.11246897

Modular Decision-Making and Drivable Areas for Multi-Agent Autonomous Racing

Alessandro Toschi[1], Francesco Prignoli[1] and Marko Bertogna[1]

***Abstract*— This paper presents an interaction-aware, modular framework for local trajectory planning in autonomous driving, particularly suited for multi-agent racing scenarios. Our framework first identifies viable drivable areas (tunnels), taking into account predictions of other agents' behaviors, and subsequently utilizes a high-level decision-making module to select the optimal corridor considering both static and moving vehicles. This decision-making module also strategically determines when to follow an opponent or initiate an overtaking maneuver, while ensuring compliance with racing regulations. A Model Predictive Control (MPC) module is then employed to compute an optimal, collision-free trajectory within the chosen corridor. The proposed modular architecture simplifies the computational complexity typically associated with MPC optimization and facilitates independent component testing. Simulations and real-world tests on various racing tracks demonstrate the efficacy of our approach, even in highly dynamic interactive scenarios with multiple simultaneous opponents; videos of these and additional experiments are available at https://atoschi.github.io/tunnels-framework/.**

I. INTRODUCTION

Planning is a pivotal component in autonomous racing, particularly within multi-agent scenarios characterized by competitive dynamics. Despite many available planning and control methodologies, managing interactions among competing agents remains an open challenge in autonomous driving. Racing differs significantly from urban driving because it requires aggressive maneuvers near the limits of vehicle dynamics, demanding specialized strategies for high-performance and rapid decision-making. Both racing and standard autonomous driving impose regulatory constraints to reduce collisions. To minimize collision risks, planners must adhere to these rules while implicitly cooperating, even in competitive settings. Racing planners must balance multi-agent interactions, account for their own objectives, and leverage predictions from forecasting modules. Consequently, innovations derived from racing scenarios hold significant potential for broader autonomous driving contexts when adapted to more conservative safety and regulatory standards. Typically, planning in autonomous racing employs a hierarchical strategy, comprising global and local planners. The global planner calculates an optimal raceline offline, optimizing for minimal lap times, whereas the local planner dynamically generates collision-free trajectories in real-time, adapting both the racing line and the associated speed profile as needed, while adhering to track conditions and signals. This bifurcation leverages extensive predictive capabilities and precomputed optimization solutions to enhance overall performance. Details about the global planner and the autonomous vehicle used in these tests can be found in [1].

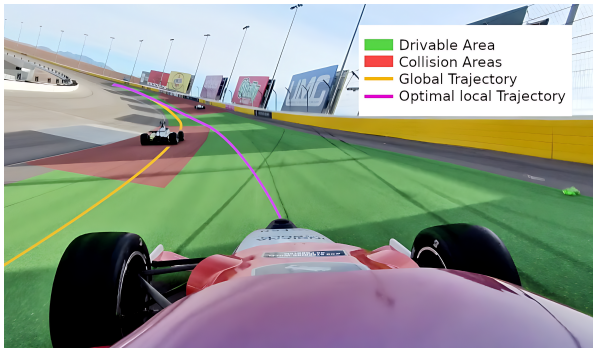

Fig. 1. Planning framework illustrative representation: onboard of Dallara AV-24 - UNIMORE Racing overtaking two vehicles during testing of the Indy Autonomous Challenge at CES 2025 at the Las Vegas Motor Speedway.

In this paper, we introduce a modular framework for local trajectory planning in autonomous racing. Our framework first identifies viable drivable regions (tunnels) and subsequently employs a high-level decision-making module to select the optimal corridor based on the immediate racing scenario, deciding not only lateral positioning (left or right relative to opponents) but also when to maintain a following distance and when to initiate overtaking maneuvers. A Model Predictive Control (MPC) based planner is then utilized to compute an optimal trajectory within the selected corridor as depicted in a conceptual representation in Fig.1. This modular approach reduces MPC's computational complexity, avoids non-convexity and local minima issues, and enables independent testing and efficient debugging.

The next section reviews pertinent literature and highlights the contributions of this study. Section III outlines the architecture and algorithms of the framework, while Sections IV-V provide and analyze both simulation and empirical results to showcase the effectiveness of the proposed approach.

II. RELATED WORK

Local trajectory planning has been widely explored in the literature, but in autonomous racing applications, particularly those tested on hardware [2], the diversity of applied planning techniques is considerably narrower.

Neural network-based planners have demonstrated performance exceeding that of human experts in simulation [3], but their effectiveness in real-world racing scenarios remains largely unverified [4], [5], [6]. In contrast, sampling-based

[1]University of Modena and Reggio Emilia, Italy,
{alessandro.toschi, francesco.prignoli,
marko.bertogna}@unimore.it

[7], [8] and graph-based [9], [10] approaches, or hybrid combinations of both [11], [12], remain the most commonly employed techniques for full-scale autonomous racing. These methods are well-established, computationally predictable, and effective in structured scenarios. However, they exhibit limitations: overtaking maneuvers are often restricted to straight sections of the track [13], or require additional physical constraints to be explicitly imposed [14], as the original sampling algorithm does not inherently account for them, leading to enforced limitations within the planner and necessitating further smoothing steps to fully exploit vehicle dynamics. MPC is widely considered a strong candidate for racing due to its predictive capabilities and ability to incorporate dynamic constraints [15]. Various MPC-based strategies have been explored, including combinations with game theory, both for full-scale [16] and small-scale [17] autonomous cars, as well as approaches that exploit track layout characteristics [18]. Additionally, nonlinear MPC has been demonstrated to be effective (in simulation), even when integrating decision-making directly within the optimization process [19], [20]. However, incorporating complex collision-avoidance mechanisms or dynamically changing track-bound constraints within MPC can make real-time performance challenging.

To mitigate these challenges, our framework follows a modular approach inspired by [21] where drivable area computation is decoupled from trajectory optimization. However, in that work, opponent predictions are not considered, as it was tested on static small-scale vehicles. State-machine-based solutions [22] can be effective when dealing with a single opponent but do not scale well in multi-agent settings, where the number of transitions increases significantly, making the system prone to errors. More recent works propose hierarchical planning architectures [23], leveraging MPC for merging precomputed racing lines [24] or incorporating learned opponent behavior into planning decisions [25]. However, none of these methods explicitly address how multiple dynamic interactions can be considered simultaneously, especially in real-time experiments. This paper addresses this gap by demonstrating the effectiveness of a modular framework that combines drivable area computation and high-level decision-making to handle multi-agent interactions dynamically, even at high speeds and accelerations, while ensuring compliance with externally imposed rules. Furthermore, the proposed architecture is designed to be flexible and extensible, providing a foundation for future advancements in autonomous racing planning.

III. PLANNING FRAMEWORK

The proposed planning workflow is illustrated in Fig.2 and follows a classical *Perceive-Plan-Act* structure. The *Perception* stage provides environment constraints, detected opponents and their predicted behavior, which are then processed in the *Planning* stage to generate a feasible trajectory. Finally, the *Actuation* stage executes the selected trajectory via the vehicle control system. At the core of the framework lies the *Tunnel Framework*, which computes the drivable

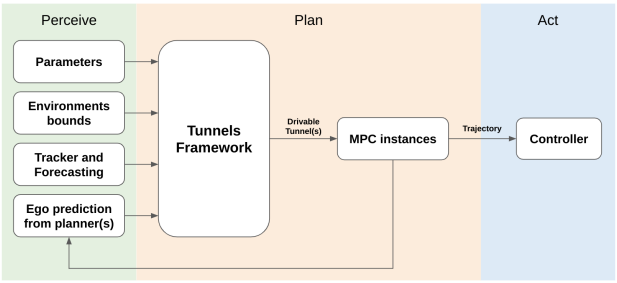

Fig. 2. Simplified block scheme of planning modules inside the autonomous stack.

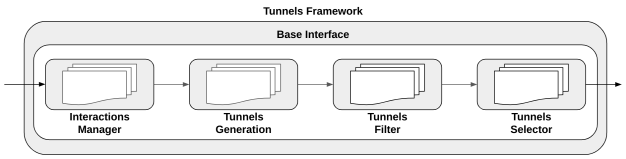

Fig. 3. Tunnels framework: pipeline divided into four blocks.

areas and selects the best option when multiple choices exist. These *Tunnels* are regions in Frenet coordinates, derived by deforming the given *Environment Bounds*. The Frenet frame is especially useful because it separates longitudinal progress from lateral position, allowing track boundaries to be directly expressed as state constraints in the MPC. The framework operates based on *Parameters* categorized into three main groups:

- Overtaking and obstacle avoidance: tunnel size, safety margins relative to vehicles and environment bounds, and overtaking aggressiveness.
- Being overtaken: space allocation for other agents and thresholds to recognize yielding scenarios.
- External signals: behavior directives from the autonomous stack, such as overtaking permissions (e.g., green flags in racing contexts) or Adaptive Cruise Control (ACC) following distances.

The *Tracker* module assigns unique identifiers to detected opponents, ensuring consistency across consecutive detections, while the *Forecasting* module predicts opponent trajectories. Although the details of forecasting are outside the scope of this work, it is important to highlight that predicted behaviors are influenced by opponent interactions via a dedicated planner for each agent. This ensures that forecasting takes into account mutual collision avoidance between opponents, rather than treating them as independent moving obstacles. The *MPC Instances* operates both as an input and an output: they use tunnel margins as constraints while also leveraging the prediction horizon to refine ego-vehicle trajectory forecasting. The proposed framework has been validated with both single and multiple MPC instances, as detailed in III-D and demonstrated in IV. The framework consists of four main blocks, as illustrated in Fig.3, each representing a key stage of the planning pipeline:

1) *Interactions Manager*: Identifies opponents interactions and determines track occupancy constraints.

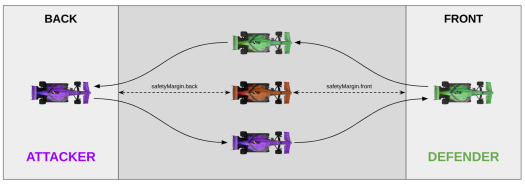

Fig. 4. Attacker-Defender flag: hysteresis-based classification of opponent status.

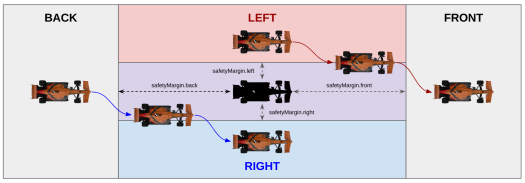

Fig. 5. EgoLoc: Structure of four flags indicating ego vehicle (orange) relative position to opponents (black).

2) *Tunnels Generator*: constructs potential drivable corridors based on opponents interactions.
3) *Tunnels Filter*: validates and refines the generated tunnels, enforcing feasibility constraints.
4) *Tunnels Selector*: selects the optimal tunnel(s) for trajectory optimization, using single or parallel selection strategies.

These blocks are implemented as C++ virtual classes, ensuring a standardized interface while allowing flexible customization. New implementations can be seamlessly integrated, and thanks to a modular design, users can configure the system at runtime without recompilation. The subsequent subsections provide a detailed explanation of each component.

A. Interactions Manager

The *Interactions Manager* is the first block in the pipeline. It interpolates track boundaries, ego-vehicle trajectories, and opponent predictions over a shared time horizon. For each detected agent, it generates vectors of boolean interaction flags indicating longitudinal overlaps, obtained by comparing the ego prediction with the forecasted one, while accounting for vehicle dimensions and safety margins. These interaction points determine how tunnels deviate from the nominal environment bounds.

Additional interaction-based flags include:

- *Attacker-Defender*: determines whether an opponent is overtaking or being overtaken, using a positional hysteresis inside the longitudinal safety margins (Fig.4).
- *EgoLocs*: identifies the ego-vehicle's relative position with respect to each opponent. Safety margins define four discrete regions (Fig.5), where the purple zone accounts for hysteresis.
- *RoWs (Right-of-Way Flags)*: determines whether the ego-vehicle has priority or should yield to specific opponents.

At this stage, only the *track tunnel* (an interpolated segment of tightened track bounds) is computed. These interaction flags are then passed to subsequent modules to refine drivable area estimation and decision-making.

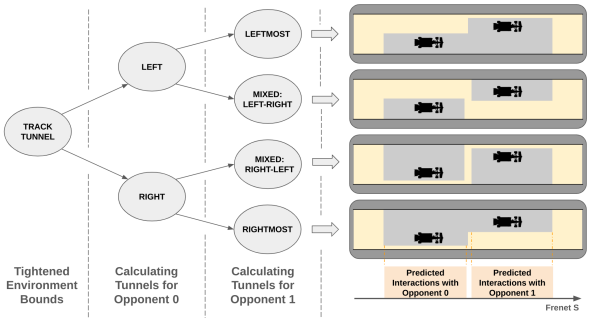

Fig. 6. Recursive tunnel generation: an example with two static opponents resulting in four tunnels. The number of calculated tunnels doubles at each iteration.

B. Tunnels Generator

The *Tunnels Generator* constructs all possible drivable corridors, potentially limiting their longitudinal extent. For N detected opponents, a total of 2^N tunnels are generated recursively. The base tunnel is refined iteratively, applying lateral safety margins for each opponent to create left and right tunnel variants. This results in two extreme tunnels (leftmost and rightmost) and $2^N - 2$ mixed tunnels that accommodate overtaking on either side (Fig.6 - Algorithm 1). The *attacker-defender* flags influence lateral margin sizing and determine whether longitudinal constraints should be imposed in scenarios where overtaking is prohibited or unsuitable. The generated tunnels are then forwarded for feasibility validation.

Algorithm 1: Recursive Tunnel Generation

Input: List of base tunnels (starting from track tunnel)
Output: Updated list of tunnels

calcTunnels$(baseTunnels)\{$
if $size(baseTunnels) = nTotalTunnels$ **then**
  return;
$calculatedTunnels = baseTunnels$;
if $opponent.defender$ **then**
  $calculatedTunnels = applyMarginsToOvertake$;
else
  $calculatedTunnels = applyMarginsToLetSpace$;
calcTunnels$(calculatedTunnels)$;
$\}$

C. Tunnels Filter

The *Tunnels Filter* refines tunnel bounds and assigns per each one an *allowed* flag indicating feasibility. Adjustments ensure that tunnels remain within track limits and do not intersect. Recursive generation may produce overlapping margins in constrained environments (e.g., near track edges or densely packed opponents), which are then clamped to the environment bounds and assigned a minimum width. Factors influencing tunnel feasibility include:

- Ego position: *egoLocs* disable tunnels that are opposite to the ego-vehicle's current side relative to an opponent.
- Tunnel width: if a tunnel's width falls below a parameterized threshold, it is disallowed unless the *RoWs* flags indicate that an opponent is expected to yield.

When a tunnel is only partially or not feasible, a longitudinal constraint ensures the ego-vehicle stays behind the first opponent that cannot be safely passed.

D. Tunnels Selector

The *Tunnels Selector* determines which tunnels are used for trajectory optimization. Two implementations exist:

1) Single Selector: A single tunnel is selected for optimization, requiring only one MPC instance. If no *allowed* tunnels exist, a constrained fallback tunnel is enforced to ensure obstacle avoidance and safe deceleration. An initial trajectory is computed using a hyperbolic tangent approach, serving as a rough curvature estimate to gauge how far we deviate from the racing line. The evaluation of the tunnel's cost (C_{tunnel}), as shown in Eq.1, primarily considers the drivable area dimension cost (C_A), along with the estimated curvature cost (C_{traj}):

$$C_{\text{tunnel}} = C_{\text{prev}} + C_A \cdot C_{\text{side}} + C_{\text{traj}}. \tag{1}$$

In this formula, C_{prev} weighs the prior choice to make it unlikely that the overtaking side will be altered unless a tunnel becomes *not allowed*. The factor C_{side} serves as a multiplier on the area cost, increasing emphasis on mixed tunnels; the more the side alterations, the greater the impact.

2) Parallel Selector: Two tunnels are selected, requiring at least two MPC instances. The selection prioritizes the outermost *allowed* tunnels on each side. As with the *Single Selector*, cost-based criteria are used, but here each MPC instance produces its own trajectory. The final decision is made by comparing just the cost of these MPC-generated trajectories and the previous choice, allowing model-based reasoning to select the most suitable path.

IV. RESULTS

This section presents results from static validation of decision-making and tunnel generation to experimental on-track tests. Verifying that the planner avoids obstacles or overtakes without collisions is straightforward. However, assessing whether a maneuver was the best possible choice is more challenging, as no universal metric exists to evaluate its optimality. The choice depends not only on the available space but also on the cross-dependent behaviors of all the agents. To demonstrate the effectiveness of the proposed approach, we present a set of representative cases that illustrate the decision-making process in different scenarios.

A. Static Test

A static test provides precise control by positioning opponents with set distances and velocity deltas while injecting artificial trajectory predictions. Unlike full simulations, it isolates a single iteration of the *Tunnels Framework* as an intermediate step before full validation. Its key advantage lies in their deterministic, repeatable nature, facilitating fine-tuning of cost functions and analyzing prediction effects on tunnel generation and decision-making.

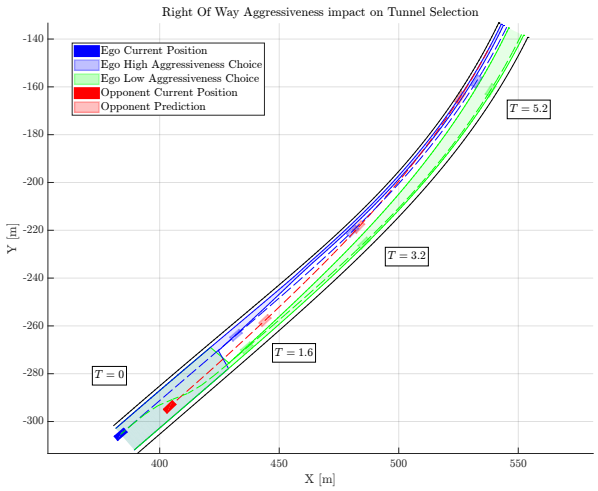

Fig. 7. Overlapping of two static tests. The blue tunnel and ego vehicle is the result of the test once the overtake aggressiveness is set to the maximum value, the green tunnel and vehicle once the aggressiveness is set to the minimum

For instance, Fig. 7 illustrates the outcome of two static tests performed using the *Single Selector*. In this scenario, a single opponent is positioned 23 m in front of the ego vehicle. Both vehicles are entering a corner, but the ego vehicle has a speed advantage of 15% higher than the opponent. The two tests were conducted with identical configurations, except for a variation in the right-of-way aggressiveness parameter. While this parameter does not affect how tunnels are initially computed, it influences tunnel selection. Referring to Eq. 1, in a static test, the cost of the previously selected tunnel is null, and the C_{side} has no influence as there are no mixed tunnels due to the presence of only one opponent. The right tunnel (green) has a larger area, leading to a lower cost. However, its trajectory (dashed green line in Fig. 7) deviates significantly from the racing line (dashed blue line), increasing curvature and overall cost. In contrast, the left tunnel has a slightly higher cost due to its smaller area but is more favorable in this case, as the racing line remains inside it where the tunnel narrows, eliminating additional curvature costs. The *right-of-way aggressiveness* parameter affects how opponent predictions influence tunnel selection. A lower setting considers only the current positions, verifying whether the ego vehicle is already within right of way margins. A higher setting evaluates the full prediction horizon, determining if the tunnel will narrow at a point where the other agent should already yield. Intermediate values progressively incorporate parts of the prediction. In the tests shown in Fig. 7, the right-of-way distance is set to 10 m, smaller than the initial gap between vehicles at $T = 0$ s. With low aggressiveness, the planner selects the right tunnel (green) since the left tunnel is too narrow and thus *not allowed*. At $T = 1.6$ s, a predicted interaction suggests that vehicles will enter the right-of-way margin, so

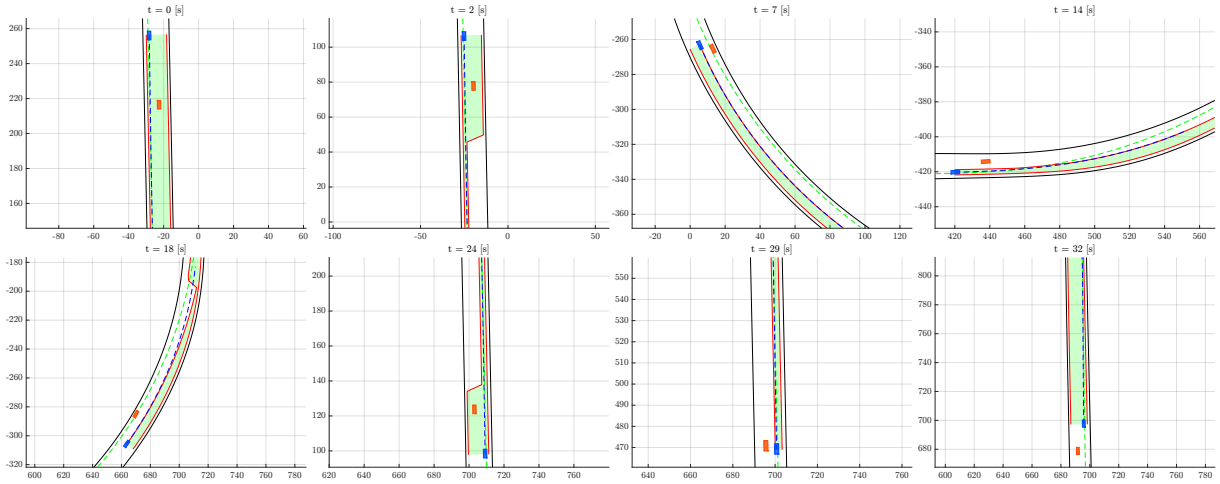

Fig. 8. Time sequence of an high speed overtake at Indianapolis Motor Speedway during the Indy Autonomous Challenge. The ego vehicle approached the opponent at 267 km/h (t = 0 s), slowed down to 249 km/h due to the turn (t = 18 s), and finally completed the overtake at 276 km/h (t = 32 s). Telemetry and vehicle cameras video available at https://youtu.be/xHNL2yvLBUM.

with high aggressiveness, the planner assumes the opponent will yield space, keeping the left tunnel allowed. At $T = 3.2$ s, if the opponent does not yield, the left tunnel would lead to a collision, illustrating how careful parameter tuning is required to balance rule compliance and trust in opponent behaviors.

B. Experimental Validation

The current version of the *Tunnels Framework* has been experimentally tested on two tri-oval circuits and one oval track. Preliminary results on a road course will be presented in the next section. Here, we illustrate time sequences for two challenging scenarios.

1) Oval Speedway – Parallel Selector: This first example highlights the fastest overtake performed during the 2024 Indy Autonomous Challenge final at the Indianapolis Motor Speedway. In this scenario, a high-speed vehicle is overtaken using the *Parallel Selector*. A total of six MPC instances were employed: three dedicated to longitudinal dynamics and three to lateral dynamics. Each pair operated on a different tunnel; two pairs relied on tunnels from the *Parallel Selector*, while one pair continuously performed ACC. After their parallel execution, a decision-making module selected which of the three generated trajectories to forward to an LQR controller.

Among the numerous overtakes performed during testing and competition, this case is particularly instructive from a decision-making standpoint. The maneuver is initiated but unintentionally aborted, as taking the outside line into the turn proved slower. It also highlights the planner's ability to respect track boundary constraints even near the vehicle's dynamic limits. The key phases of the overtake are illustrated in Fig. 8 and described as follows:

- *t = 0 s*: The ego vehicle approaches on the main straight with no predicted interactions.
- *t = 2.5 s*: The system predicts interactions and initiates an outside overtake instead of maintaining ACC, as it deems this approach feasible.
- *t = 7 s*: The overtake continues into Turn 1 while closing in on the opponent.
- *t = 14 s*: The opponent speed is higher through the turn, creating a gap; the vehicles remain within longitudinal safety margins, so the tunnel remains tightened.
- *t = 18 s*: Both cars are still in Turn 2, with the gap slowly increasing. The ego vehicle is near the track boundary to respect constraints, staying away from the racing line but within the vehicle's dynamic limits.
- *t = 24 s*: Entering the back straight, the car moved beyond the back safety margin, yet further interactions are predicted as the ego vehicle accelerates.
- *t = 29 s*: The vehicles are side by side along the back straight, with the ego car proceeding to overtake.
- *t = 32 s*: At the end of the back straight, the overtake is completed, and the entire track becomes available again.

2) Tri-Oval Speedway – Single Selector: This second example uses the *Single Selector* and thus a single MPC instance, referred to as MPC-CURV (see [26]). Although this scenario is less challenging in terms of speed, it involves four vehicles interacting on track during a trial race for the Indy Autonomous Challenge at CES 2025. Each vehicle has a limited *Push-to-Pass (P2P)* budget, which grants a temporary speed boost for a few seconds, followed by a cooldown period. Since both the ego vehicle and its opponents can activate P2P at unpredictable times, longitudinal predictions inevitably carry some error. Figure 9 illustrates how the ego vehicle performs a double overtake in this uncertain context. The key phases of the overtake are:

- *t = 0 s*: Green flags were issued a few seconds earlier. The ego vehicle detects two opponents (almost side by side) ahead and one behind. No feasible tunnel is found, so it remains on the racing line and employs ACC to maintain safe longitudinal distances.
- *t = 9 s*: Approaching Turn 1 with P2P still active but ending soon, the ego vehicle identifies a slower

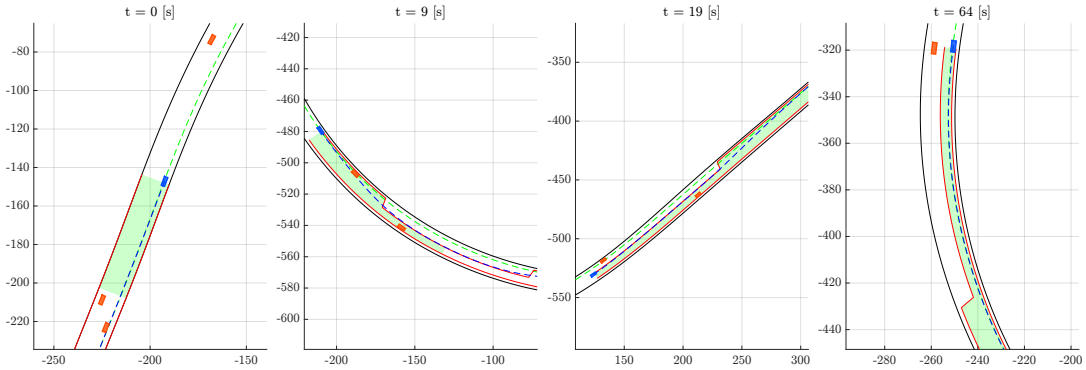

Fig. 9. Time sequence of a double overtake at Las Vegas Motor Speedway during testing Indy Autonomous Challenge testing at CES 2025. Onboard video available at https://www.youtube.com/watch?v=7in41qI9jg8.

opponent on the inner lane. The planner selects the right tunnel but imposes a longitudinal limit to follow the second opponent while overtaking the nearest one. This moment represents a few seconds before the onboard screenshot shown in Fig.1.

- *t = 19 s*: The previously predicted speeds prove inaccurate; by now, the ego vehicle was expected to have cleared the slower opponent. Exiting Turn 2 on the backstretch, the ego reactivates P2P, accelerating at 6 m/s^2. Only one opponent remains relevant to the overtaking tunnel, as the other is moving faster and does not interfere.
- *t = 64 s*: After one full lap and several alternating P2P activations between attacker and defender, the ego vehicle completes the second overtake entering Turn 1.

Table I reports the performance profiling of the *Tunnels Framework* running on dSPACE AUTERA equipped with an Intel Xeon Processor D-2166NT (2.3 GHz) CPU. Although tunnel generation grows exponentially with the number of opponents, the planner operates at 20 Hz, and combined decision-making plus tunnel generation remains below 1 ms, about 2% of the total available cycle. This leaves ample time for the MPC instance to compute the required trajectories in real time.

Agents	Samples	Median$_{ET}$	Max$_{ET}$	Min$_{ET}$
0	15146	0.24	0.39	0.19
1	22402	0.36	0.61	0.19
2	8950	0.53	0.89	0.29
3	683	0.86	1.01	0.42

TABLE I
DSPACE PROFILATION: EXECUTION TIMES STATISTICS IN MS

C. Road Course Simulation

Although the presented solution has not yet been tested on a real road course, its adaptability to such circuits has been demonstrated through multiple simulation platforms. In particular, extensive tests were carried out on the Yas Marina circuit using the Super Formula EAV24 vehicle model, with several opponents sharing the track. The effectiveness of the proposed method in managing multi-agent interactions under varied conditions can be observed in the accompanying simulation (https://atoschi.github.io/tunnels-framework/).

To obtain a comprehensive evaluation, the proposed framework was tested across three distinct simulators, each with specific strengths and limitations. The first, a multibody simulator, offered highly accurate vehicle dynamics but no interactive behavior, as opponents ignored the ego vehicle and collisions were not modeled. The second, Autoverse, developed by Autonoma.ai, improved infrastructure support by enabling multi-user sessions via a shared server; however, virtual opponents still failed to react to the ego vehicle, limiting behavioral interactivity. Finally, Assetto Corsa, used via the interfaces described in [27], featured virtual opponents that could react both to the ego vehicle and to each other. This made it easier to tune their performance and aggression levels, though at the cost of a vehicle model that was less realistic than the one in the multibody simulator.

Overall, these simulation environments confirmed the flexibility and robustness of the proposed approach across various levels of fidelity. Despite the different limitations and strengths of each simulator, the *Tunnels Framework* consistently demonstrated its ability to handle complex interactions, laying a strong foundation for future testing on actual road courses.

V. CONCLUSION

We have shown that the proposed framework handles multi-agent interactions by systematically identifying and updating drivable corridors, ensuring real-time adaptability and computational efficiency. While initially developed for MPC, it could be integrated with other planning methodologies if environment-specific bounds are provided. Validation on multiple circuits confirms its robustness under varied conditions. Nevertheless, assessing complex maneuvers is subjective: overtaking on one side may seem advantageous, yet the opposite side might prove better, and in some cases, simply maintaining position is the safest decision. In simulation, one can employ ideal detection and forecasting to isolate the planner's performance. However, real-world testing inevitably depends on additional modules in the autonomous driving stack, including the forecasting component, which

remains a critical element for achieving near free-racing conditions. Predicting behavior grows increasingly difficult as the number of interacting agents rises, making strict rules essential - much like in human motorsports - to enhance predictability and minimize unforeseeable actions. Similarly, in urban driving contexts, rules and regulations exist but cannot guarantee perfect mutual understanding between drivers. Simulation thus remains an indispensable tool for creating robust solutions and testing a wide array of edge cases. Building on these considerations, our future work includes:

- Quantitative comparison between *Single* and *Parallel Selector*, and other planning approaches.
- Automating the tuning of overtaking margins and aggressiveness, leveraging known errors in detection, forecasting, and controller tracking based on empirical data.
- Introducing probabilistic elements in tunnel selection to weigh both the likelihood of success and collision risk.
- Developing a *Selector* informed by learning-based methods, trained on synthetic or human-derived simulation data, further improving the adaptability to complex scenarios.

REFERENCES

[1] A. Raji, *et al.*, "er.autopilot 1.1: A Software Stack for Autonomous Racing on Oval and Road Course Tracks," *IEEE Transactions on Field Robotics*, vol. 1, pp. 332–359, 2024. DOI: 10.1109/TFR.2024.3501252

[2] J. Betz, *et al.*, "Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 458–488, 2022, arXiv:2202.07008 [cs]. DOI: 10.1109/ojits.2022.3181510

[3] P. R. Wurman, *et al.*, "Outracing champion Gran Turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, Feb. 2022. DOI: 10.1038/s41586-021-04357-7

[4] C. Chung, H. Seong, and D. H. Shim, "Learning from Demonstration with Hierarchical Policy Abstractions Toward High-Performance and Courteous Autonomous Racing," Nov. 2024, arXiv:2411.04735 [cs]. DOI: 10.48550/arXiv.2411.04735

[5] R. Trumpp, E. Javanmardi, J. Nakazato, M. Tsukada, and M. Caccamo, "RaceMOP: Mapless Online Path Planning for Multi-Agent Autonomous Racing using Residual Policy Learning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Abu Dhabi, United Arab Emirates: IEEE, Oct. 2024, pp. 8449–8456. DOI: 10.1109/IROS58592.2024.10801657

[6] A. Raji, *et al.*, "Motion Planning and Control for Multi Vehicle Autonomous Racing at High Speeds," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. Macau, China: IEEE, Oct. 2022, pp. 2775–2782. DOI: 10.1109/ITSC55140.2022.9922239

[7] S. Thrun, *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, Sept. 2006. DOI: 10.1002/rob.20147

[8] C. Jung, *et al.*, "An Autonomous System for Head-to-Head Race: Design, Implementation and Analysis; Team KAIST at the Indy Autonomous Challenge," Mar. 2023, arXiv:2303.09463 [cs]. DOI: 10.48550/arXiv.2303.09463

[9] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp, "Multilayer Graph-Based Trajectory Planning for Race Vehicles in Dynamic Scenarios," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand: IEEE, Oct. 2019, pp. 3149–3154. DOI: 10.1109/ITSC.2019.8917032

[10] D. Lee, C. Jung, A. Finazzi, H. Seong, and D. H. Shim, "A Resilient Navigation and Path Planning System for High-speed Autonomous Race Car," Sept. 2022, arXiv:2207.12232 [cs]. DOI: 10.48550/arXiv.2207.12232

[11] L. Ögretmen, M. Rowold, M. Ochsenius, and B. Lohmann, "Smooth Trajectory Planning at the Handling Limits for Oval Racing," *Actuators*, vol. 11, no. 11, p. 318, Nov. 2022. DOI: 10.3390/act11110318

[12] L. Ögretmen*, M. Rowold*, T. Betz, A. Langmann, and B. Lohmann, "A Hybrid Trajectory Planning Approach for Autonomous Rule-Compliant Multi-Vehicle Oval Racing," *SAE International Journal of Connected and Automated Vehicles*, vol. 7, no. 1, pp. 12–07–01–0007, Sept. 2023. DOI: 10.4271/12-07-01-0007

[13] Y. Na, *et al.*, "AutoKU: An Autonomous Driving System Design for the World's First Mass-Produced Vehicle in Multi-Vehicle Racing Environment," in *2024 IEEE Intelligent Vehicles Symposium (IV)*. Jeju Island, Korea, Republic of: IEEE, June 2024, pp. 1373–1380. DOI: 10.1109/IV55156.2024.10588679

[14] L. Ögretmen, M. Rowold, A. Langmann, and B. Lohmann, "Sampling-Based Motion Planning with Online Racing Line Generation for Autonomous Driving on Three-Dimensional Race Tracks," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, June 2024, pp. 811–818, arXiv:2403.18643 [cs]. DOI: 10.1109/IV55156.2024.10588726

[15] Ayoub Raji, "Model predictive planning and control for autonomous racing, from HPC to embedded platforms," Ph.D. dissertation, Università degli Studi di Parma, Parma, Italy, 2024.

[16] C. Jung, S. Lee, H. Seong, A. Finazzi, and D. H. Shim, "Game-Theoretic Model Predictive Control with Data-Driven Identification of Vehicle Model for Head-to-Head Autonomous Racing," June 2021, arXiv:2106.04094 [cs]. DOI: 10.48550/arXiv.2106.04094

[17] A. Liniger, "Path Planning and Control for Autonomous Racing," Ph.D. dissertation, ETH Zurich, 2018, artwork Size: 136 p. Medium: application/pdf Pages: 136 p. DOI: 10.3929/ETHZ-B-000302942

[18] J. Bhargav, J. Betz, H. Zheng, and R. Mangharam, "Track based Offline Policy Learning for Overtaking Maneuvers with Autonomous Racecars," July 2021, arXiv:2107.09782 [cs]. DOI: 10.48550/arXiv.2107.09782

[19] A. Buyval, A. Gabdulin, R. Mustafin, and I. Shimchik, "Deriving overtaking strategy from nonlinear model predictive control for a race car," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC: IEEE, Sept. 2017, pp. 2623–2628. DOI: 10.1109/IROS.2017.8206086

[20] N. Li, E. Goubault, L. Pautet, and S. Putot, "A Real-Time NMPC Controller for Autonomous Vehicle Racing," in *2022 6th International Conference on Automation, Control and Robots (ICACR)*. Shanghai, China: IEEE, Sept. 2022, pp. 148–155. DOI: 10.1109/ICACR55854.2022.9935523

[21] A. Liniger, A. Domahidi, and M. Morari, "Optimization-Based Autonomous Racing of 1:43 Scale RC Cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, Sept. 2015, arXiv:1711.07300 [math]. DOI: 10.1002/oca.2123

[22] V. Suresh Babu and M. Behl, "Threading the Needle—Overtaking Framework for Multi-agent Autonomous Racing," *SAE International Journal of Connected and Automated Vehicles*, vol. 5, no. 1, pp. 12–05–01–0004, Jan. 2022. DOI: 10.4271/12-05-01-0004

[23] G. Jank, M. Rowold, and B. Lohmann, "Hierarchical Time-Optimal Planning for Multi-Vehicle Racing," Sept. 2023, arXiv:2309.06768 [cs]. DOI: 10.48550/arXiv.2309.06768

[24] A. Ticozzi, G. Panzani, M. Corno, and S. M. Savaresi, "Optimal smooth polynomial lane change generation for autonomous racing," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 11 834–11 840, 2023. DOI: 10.1016/j.ifacol.2023.10.584

[25] N. Baumann, *et al.*, "Predictive Spliner: Data-Driven Overtaking in Autonomous Racing Using Opponent Trajectory Prediction," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 1816–1823, Feb. 2025, arXiv:2410.04868 [cs]. DOI: 10.1109/LRA.2024.3519878

[26] A. Raji, *et al.*, "A Tricycle Model to Accurately Control an Autonomous Racecar with Locked Differential," in *2023 IEEE 11th International Conference on Systems and Control (ICSC)*, Dec. 2023, pp. 782–789, arXiv:2312.14808 [eess]. DOI: 10.1109/ICSC58660.2023.10449744

[27] A. Remonda, *et al.*, "A Simulation Benchmark for Autonomous Racing with Large-Scale Human Data," July 2024, arXiv:2407.16680 [cs]. DOI: 10.48550/arXiv.2407.16680